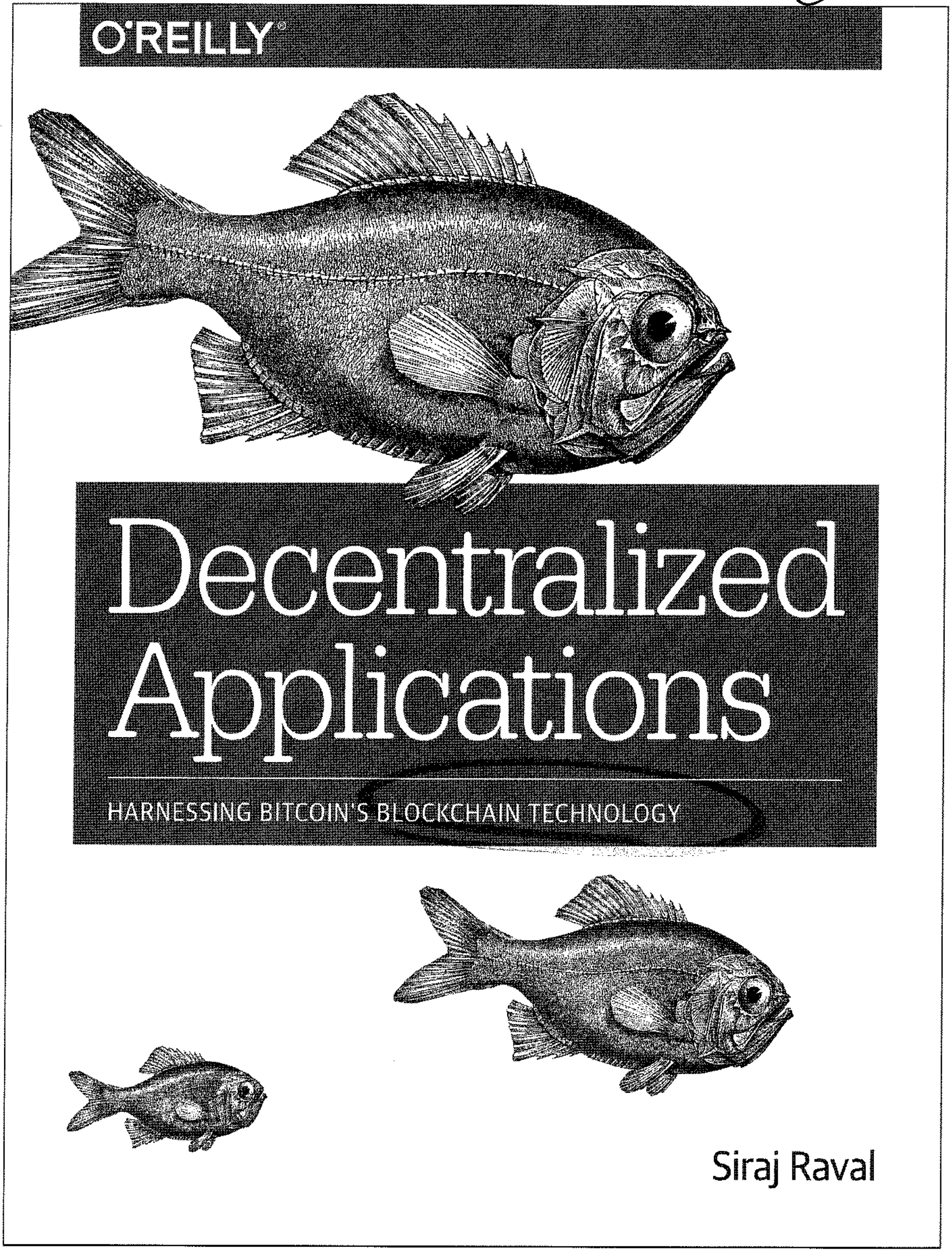
EXHIBIT A

2



O'REILLY

Decentralized Applications

HARNESSING BITCOIN'S BLOCKCHAIN TECHNOLOGY

Siraj Raval

CHAPTER 1

What Is a Decentralized Application?

A new model for building massively scalable and profitable applications is emerging. Bitcoin paved the way with its cryptographically stored ledger, scarce-asset model, and peer-to-peer technology. These features provide a starting point for building a new type of software called *decentralized applications*, or *dapps*. Dapps are just now gaining media coverage but will, I believe, someday become more widely used than the world's most popular web apps. They are more flexible, transparent, distributed, resilient, and have a better incentivized structure than current software models. This is the first book that will help you to understand them and create your own.

Preliminaries: What Is Bitcoin?

Before we get into the details of dapps, let's talk a little about Bitcoin and the Web. We've seen the Web grow considerably, by orders of magnitude, over the past decade. Billions of people are coming online as Internet-connected device distribution expands globally. At first glance, the standards of communication set in the Internet Protocol Suite seem to be working well enough: the Link Layer puts some data on a wire; the Internet Layer routes the data; the Transport Layer persists the data; and the Application Layer delivers abstractions of the data in the form of applications. All four protocols work together seamlessly for exchanging data, but not value. Bitcoin acts as a fifth protocol layer for value transfer that lives up to the standards of the other four.

We do have an existing way of sending payments on the Web. The problem is that they all involve inefficient legacy systems like Automated Clearing House (ACH) that were designed before the Internet. These traditional payment systems are painfully slow because they require a centralized clearing house. Machines shouldn't have to wait days for a payment to clear; they are constantly communicating with one another. They should be able to send billions of micropayments to each other to meter resources like electricity and storage space and not have to worry about the hefty transaction fees of a middleman. Bitcoin helps solve this problem.

With the advent of Bitcoin, instant, decentralized, pseudonymous value transfer is finally possible. Bitcoin's anonymous creator, who used the assumed name Satoshi Nakamoto, effectively solved the *Byzantine Generals Problem*, a problem that had plagued cryptographic research for decades. To quote from the original paper (Lamport, 1982) defining the Byzantine Generals Problem: "[Imagine] a group of generals of the Byzantine army camped with their troops around an enemy city. Communicating only by messenger, the generals must agree upon a common battle plan. However, one or more of them may be traitors who will try to confuse the others. The problem is to find an algorithm to ensure that the loyal generals will reach agreement." Achieving decentralized consensus in Bitcoin meant that no longer did one party have to go through a central authority or trust the other party to share information, including information in the form of value transactions.

Bitcoin and other cryptocurrencies will help define the fifth protocol layer of the Internet, letting machines transfer value as fast and efficiently as data. Bitcoin is a useful tool for online value transfer, but its most valuable innovation is its underlying technology, the *blockchain*, that for the first time in history made decentralized consensus possible.

The blockchain is a massively replicated database of all transactions in the Bitcoin network. It uses a consensus mechanism called proof-of-work which prevents double-spending in the network—a problem that had plagued cryptographic researchers for decades. Double-spending meant a bad actor could spend the same funds twice, denying the first transaction happened.

Proof-of-work solves this problem by having *miners* in the network solve cryptographic proofs using their hardware. Miners are Bitcoin nodes that verify a transaction and check it via its blockchain history, a timestamped record of all transactions ever made in the network. Someone could theoretically alter their blockchain history, but with proof-of-work, they would also need to have the majority of computational power in the network to verify it. Because the Bitcoin network has much more computation power at this point than all of the world's supercomputers combined, an attacker would have an extremely difficult time trying to break the network.

Proof-of-work is expensive in terms of the cost of electricity and compute workload but it's the only known prevention mechanism against Sybil attacks, in which a bad actor claims to be multiple people in a network and gains resources that they shouldn't by doing so. A successful Sybil attack on the Bitcoin network would most likely result in a complete devaluation of the currency because people would no longer trust its stability. As expensive as proof-of-work is, it's the only thing that's proven to work so far on a massive scale.

So, we have this new tool called the blockchain, a massively replicated database of transactions that's able to avoid Sybil attacks. For the first time, the blockchain lets us achieve decentralized consensus without the use of a centralized server. You might be wondering what use cases this would have, and rightly so. I'm going to be devoting a good portion of the book to helping you think about all of the possibilities and ways with which you could implement them. The important bit for now is to understand that this data structure is one of many that will help you to create profitable decentralized applications.

What Is a Decentralized Application?

Most people are familiar with the term "application" as it pertains to software. A software application is software that defines a specific goal. There are millions of software applications currently in use, and the vast majority of web software applications follow a centralized server-client model. Some are distributed, and a select few novel ones are decentralized. Figure 1-1 shows a visual representation of these three models for software.

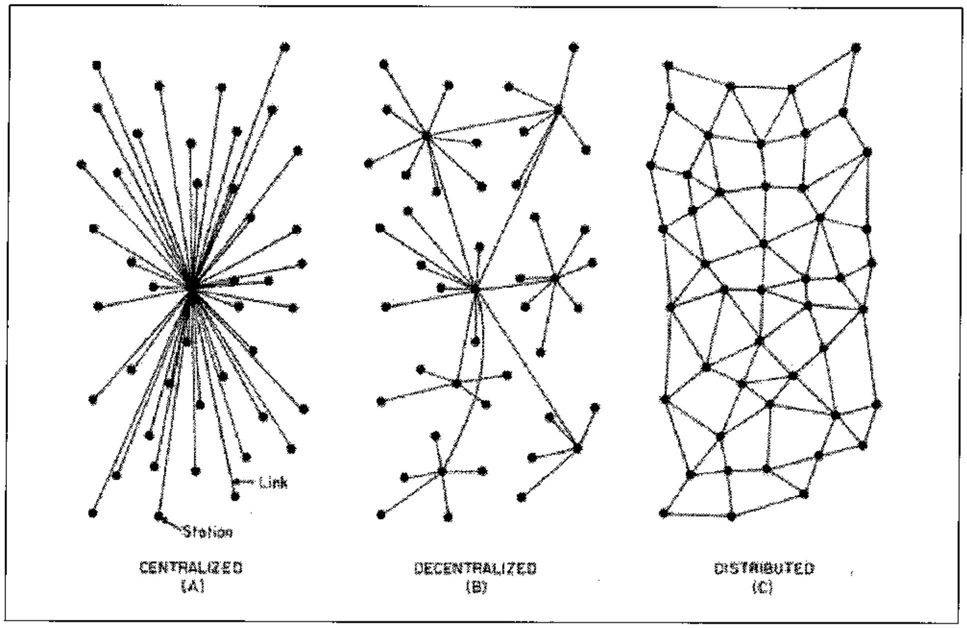


Figure 1-1. The three different types of software applications

Centralized systems are currently the most widespread model for software applications. Centralized systems directly control the operation of the individual units and flow of information from a single center. All individuals are directly dependent on the central power to send and receive information and to be commanded. Facebook, Amazon, Google, and every other mainstream service we use on the Internet uses this model. Let's call these huge services "The Stacks." The Stacks are useful because they provide a valuable service to us, but they have immense flaws that I'll go into in Chapter 2.

So, what's the difference between decentralized and distributed?

Distributed means computation is spread across multiple nodes instead of just one. Decentralized means no node is instructing any other node as to what to do. A lot of Stacks such as Google have adopted a distributed architecture internally to speed up computing and data latency. This means that a system can be both centralized and distributed.

Can a system be both distributed and decentralized?

Yes, it can. Bitcoin is distributed because its timestamped public ledger, the blockchain, resides on multiple computers. It's also decentralized because if one node fails, the network is still able to operate. That means that any app that uses a blockchain alongside other peer-to-peer tools can be distributed and decentralized.

Then, why isn't the title of this book Distributed and Decentralized Applications*?*

Centralized systems can be distributed as well. Software applications that are able to achieve decentralized consensus are a real innovation.

So, is having decentralized consensus the only requirement to being a decentralized app?

The dapp space is currently an emerging field with a lot of smart people still experimenting with new models. Different developers have different opinions on what exactly a dapp is. Some developers think that having no central point of failure is all it takes and some think that there are other requirements. The focus of this book is to talk about profitable dapps; that is, dapps from which developers and users can earn money. The reason for the profit focus is because profit is the cornerstone of a successful, robust, and sustainable dapp. Incentives keep developers building, users loyal, and miners maintaining a blockchain. To that end, Figure 1-2 shows the four features any profitable dapp should have.

Feature 1: Open Source

Decentralized, closed-source applications require users to trust that the app is as decentralized as the core developers say it is, and that they don't have access to their data through a central source. Closed-source applications thus raise a red flag to users and act as a barrier to adoption. The aversion to closed source is particularly pronounced when the application is designed to receive, hold, or transfer user funds. Although it might not be impossible to successfully launch a closed-source decentralized application, the battle would be uphill from the start, and users would favor open source competitors. Open sourcing a dapp changes the structure of its business practices so that the Internet is common denominator instead of a chain of closed silos.

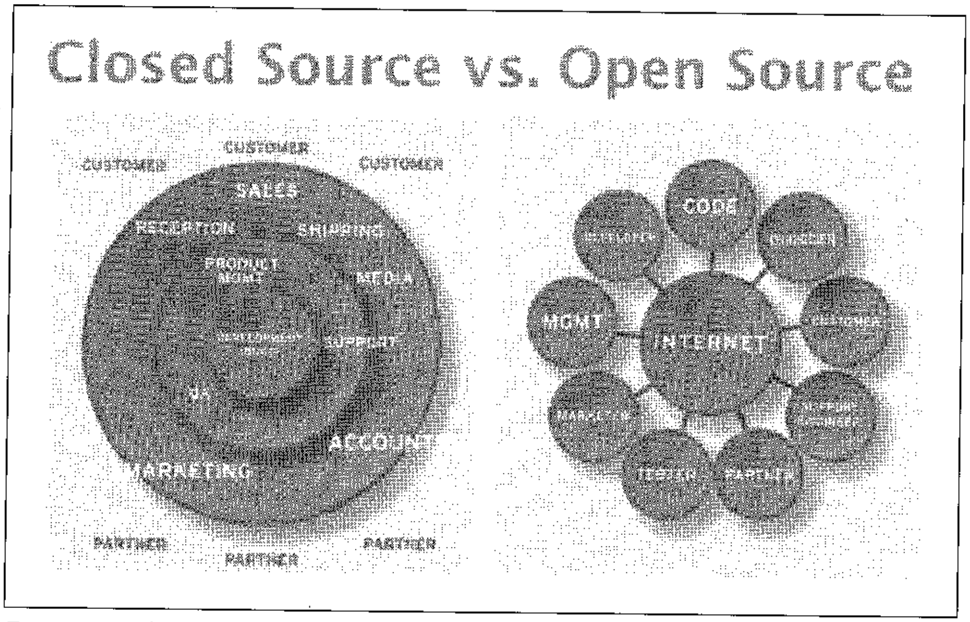


Figure 1-2. Closed source versus open source business plans

Any app can be open source. So why aren't they?

If we delve into the traditional business models, all of them require the product or service for sale to be better than that of the competitor. Open sourcing your product would mean that any competitor could take all of your work, white label it, and sell it as their own.

So, what incentive is there for app developers to open source the work from which they plan to profit?

Bitcoin is a good example of an open-source dapp from which the creator profited handsomely. Satoshi kept an initial amount of Bitcoins and let others use the rest. Because Bitcoins were limited in quantity and the network itself provided huge value to society in the form of its novel proof-of-work mechanism, the value of Bitcoin started to increase and so did his wealth. Having the app be open source made it possible for the network to achieve the transparency it needed to improve itself with developer contributions and grow trust among its users to give its coins real-world value. Open sourcing your dapp will gain the trust of potential users. Anyone can fork your dapp, but they can't fork your development team. Users want to get behind the people best suited to maintain the dapp, and often, those people tend to be the original developers.

Feature 2: Internal Currency

A question that consistently comes up in dapp circles is how to monetize a dapp. Traditional modes of monetization for centralized applications include transaction fees, advertising revenues, referral commissions, access rights to user data, and subscription services. If you open source your dapp, how are you supposed to make money? You might try programmatically inserting a fee for transactions in the network that would automatically go to the app developers' account, but that would rely on trusting users to not fork the app and take out your commission—not ideal. Neither is embedding advertising, subscription services, or any of the other centralized business models.

How is any open-source dapp developer supposed to make money?

The answer is to allocate scarce resources in the network using a scarce token: an *appcoin*. Users need this appcoin to use the network. Owners of scarce resources get paid in appcoins. In the Bitcoin network, the owners (miners) of the scarce resources (computing power) are paid with transaction fees directly from the users so that they can use the service. Because the network grew to include more users and there were a fixed amount of coins from the outset, the values of the coins grew, as well. We can apply this model to any kind of dapp. Scarce resources could be storage space, trades, images, videos, texts, ads, and so on.

Does this mean users would need to pay to use any dapp?

Yes and no. Although blockchains are pay-to-play, there are different ways to structure incentives within dapps. Users could receive a sign-up bonus of coins or even have the option to willingly sell their data or local storage space in exchange for coins. Besides using appcoins, dapp creators could monetize virtual assets like real estate in a decentralized MMORPG, domains in a special namespace, or even reputation.

Feature 3: Decentralized Consensus

Before Bitcoin, consensus on transaction validity always required some degree of centralization. If you wanted to make a payment, your transaction had to go through a clearing house that monitored all transactions. Bitcoin is peer to peer (P2P), which means nodes are able to talk to each other directly. P2P networks are not a novel thing; Distributed Hash Tables (DHTs) like BitTorrent were invented before the blockchain. DHTs are great for storing and streaming decentralized data, but if you want application-level constructs like usernames, status updates, high scores, and so forth for which you need everyone to agree on in a decentralized way, you'll also need a blockchain. The blockchain doesn't replace the need for DHTs, but it does serve to complement them. What makes the blockchain unique is that it solves the major security issue of DHTs: not forcing nodes to trust each other on the validity of data. The blockchain is a decentralized database of transactions and it's the first decentralized database that is highly tamper-resistant. The blockchain's security was a dominant design goal. It is the first ever organizationally decentralized and logically centralized transaction log. Here is a map of what I mean.

	Organizationally centralized	Organizationally decentralized
Logically centralized	Paypal	Bitcoin
Logically decentralized	Excel	Email

The blockchain's innovation is decentralized consensus. If your app needs some feature that requires everyone else to agree on something, you should use a blockchain. A simple example is a username system for which it doesn't really matter who has the "@user" username; what matters is that everyone agrees who has it. There have been lots of decentralized protocols in the past, but they all required nodes to trust one another. The blockchain is an immutable record that every node has a copy of, so no one can pretend that they too are @user. This can be done via the use of *smart contracts*.

A smart contract is a piece of code that lives in a blockchain. When a preprogrammed condition is triggered, the smart contract executes the corresponding contractual clause. You might be thinking,"What makes that different from doing something like this with Stripe's API?"

```
if (user.sendsMoney(customerID))
{
runContract();
}

func runContract()
{
 println('hello world');
}
```

One big difference: smart contracts live on a blockchain, not a server. No third-party trust is required, and there is no need to trust Stripe or a server owner. So, a more formal phrase for smart contract would be a "cryptoeconomically secured execution of code." One thing to keep in mind is that not all dapp code is a smart contract, and although smart contracts have their own specific use case, for the purposes of this discussion they will generally act as one "model" in a model-view-controller dapp architecture. We'll talk more about that in depth when I begin walking through dapp architecture.

Feature 4: No Central Point of Failure

Dapps can't be shut down, because there is no server to take down. Data in a dapp is decentralized across all of its nodes. Each node is independent; if one fails, the others are still able to run on the network. There are a number of decentralized database systems on which to build dapps that allow for this feature, such as Interplanetary File System, BitTorrent, and independent DHTs.

The History of Decentralized Applications

In its early days, the Web was obviously not as useful as it is today with the array of apps and services that do everything under the sun, but it did have a more DIY distributed feel to it. The Web was pretty decentralized from the outset. The HTTP protocol connected everyone on the planet with a computing device and an Internet connection. In the HTTP protocol guidelines, there are a set of trusted servers that translate the web address you enter into a server address. Furthermore, HTTPS adds another layer of trusted servers and certificate authorities. People would host personal servers for others to connect to, and everyone owned their data. But soon, application servers began taking off and the centralized model of data ownership as we know it today was born. Why did it happen this way?

The simple answer is because it was easy, both conceptually and programmatically. It was the easiest thing to do and it worked. One individual or group pays for maintenance of a server and profits from the users that utilize the software on it. Apps like MySpace and Yahoo! were among the first popular centralized apps. More recent apps like Uber and Airbnb decentralize the "real-world" parts of a business by providing a central and trusted data store. They are among the first to allow for participation in one moneymaking endeavor from all sides of the economy. Their decentralized business model foreshadows the development of even more decentralized apps.

As the HTTP web grew larger, a new protocol was introduced by a developer named Bram Cohen, called *BitTorrent*. BitTorrent was a protocol created as a solution to the lengthy time to download huge media files via HTTP and as an improvement on some of the P2P proposals before it, like Gnutella, Napster, and Grokster. The problem was that downloading huge files took a very long time and as the Web grew, so did the size of files that were available. Meanwhile, hard-drive space was increasing and more people were connected. BitTorrent solved this by making downloaders into uploaders, as well.

If there was a file you wanted, you would download it from not one, but multiple sources. The more popular the file, the more users who would be downloading it and subsequently uploading it, which meant you would be pulling from multiple sources. The more sources, the faster the download. Seeders were rewarded with faster download speeds, whereas leechers were punished with limited speeds. This tit-for-tat system of transferring data proved to be very useful for large media files like movies and TV shows.

BitTorrent grew and is for many the de facto way to download any sort of large media file like a game or movie. BitTorrent's speed, resilience, and reward mechanism proved to be better than HTTP for large data sets.

So, why doesn't the Web work this way?

Most likely because of HTTP's first mover advantage, its infrastructure, and all of the time and money already invested in it. There are currently active projects working on upgrading the HTTP web with BitTorrent-like technology, and they'll most likely be successful because of BitTorrent's huge value proposition. As soon as BitTorrent was introduced, developers began to use the technology to create nonprofit decentralized applications. Let's look through just a few examples of recent decentralized apps.

PopcornTime

PopcornTime uses the BitTorrent protocol to stream videos between users in real time, kind of like a Netflix for torrents. It is the worst nightmare of the Motion Picture Association of America (MPAA). No regulator can shut it down, and now everyone has access to free movies. PopcornTime proved to be a useful dapp acting as a decentralized version of Netflix. The creators claim that it has been downloaded in every single country, even the two without Internet. PopcornTime uses no internal currency and doesn't need decentralized consensus, so it had no use for a blockchain. It simply streams movies and that proved to provide a lot of value.

OpenBazaar

OpenBazaar aims to be a decentralized version of Ebay. No middleman can tell sellers what they can and can't sell or decide on the fees for using the service. It's built on the BitTorrent protocol, but the problem is that the sellers must host their own stores. They need to have their own server and leave it on in order for users to be able to see their items. Ideally sellers could just upload their store data to the network, perhaps paying a small fee, without having to worry about it. This requires a decentralized system of incentivized storage miners, which we'll cover in detail in Chapter 4. OpenBazaar uses BitTorrent's protocol for data transfer and Bitcoin as currency for transactions between sellers.

FireChat

FireChat emerged with a famous use case—the 2014 Hong Kong protests for democracy. China's infamous "Great Firewall" is notorious for blocking IP addresses for content that it deems prodemocracy or just not in its interest. The protesters feared the government would try to shut down access to various social networks to stop collaboration as is possible to do with the HTTP protocol. Instead, they used FireChat, an app that used a new feature in iOS 7 called multipeer connectivity makes it possible for phones to connect to each other directly without a third party. Because it had no central point of failure, the government would be forced to manually shut down every node, and thus the protestors were able to communicate with one another robustly.

Decentralized rebellion at its finest.

Lighthouse

We'll discuss Lighthouse in detail in Chapter 5, but it is a Bitcoin wallet embedded with a series of smart contracts. These smart contracts help pledge money to certain projects just like Kickstarter. When the project goal has been reached, it becomes possible to retrieve the funds out of the project backer's Lighthouse wallet. Pledgers can undo pledges at any point without the involvement of the project creator. Lighthouse is a great example of using the existing Bitcoin infrastructure to build your dapp. It is just a UI with some Bitcoin smart contracts built in as a wallet. It works and it builds off Bitcoin's existing user base. It has decentralized consensus, it's open source, it has no central point of failure, but it doesn't issue its own currency; rather, it uses Bitcoins. It's a useful dapp but it's not profitable for the creator.

Gems

Gems is a social-messaging app that is trying to create a more fair business model than WhatsApp. Gems is issuing its own currency and letting advertisers pay users directly with it for their data rather than acting as the middleman who profits. Users can also earn gems by referring others to the network. Gems are a meta-coin built on Bitcoin that developers also receive for developing and maintaining the software. As the Gems user base grows, so does the value of the currency. Users are incentivized to grow the network and earn money just like the developers. You can think of Gems as shares in the dapp. Gems hasn't open sourced its code, so users can't verify if they truly have no central point of failure. It's a profitable app, but in my opinion it isn't robust enough to withstand competitors who fulfill the other three criteria.

So, are there any standalone dapps that satisfy all four criteria: no central point of failure, issue their own internal currency, have decentralized consensus, and are open source?

There are plenty of cryptocurrencies that satisfy all four criteria, but cryptocurrencies aren't dapps. I'm talking about decentralized social networks, ride sharing, search engines: taking The Stacks and decentralizing them. The answer is not yet. It's possible, though—the technology exists, and as soon as a few emerge, a flurry of developers will jump on the decentralized bandwagon to make some serious money for both themselves and their users. Let's talk about some of these enabling technologies.

Enabling Technologies

I've already mentioned many of the enabling technologies during our discussion on the history of decentralized applications. Bitcoin's blockchain is, of course, of primary importance, so we'll take a deeper dive into this before considering the other enabling technologies. The blockchain helped solve the Byzantine Generals Problem. That problem asks the question, "How do you coordinate among distributed nodes to come up with some sort of consensus that is resistant to attackers trying to undermine it?" The proof-of-work algorithm and the blockchain help solve this.

When Bitcoin was created, decentralized consensus became possible. Proof-of-work isn't perfect—it is both computationally and energy expensive. There are alternative cryptocurrencies out there that solve meaningful problems, like PrimeCoin, whose miners use their compute resources to find prime numbers. In a world where Bitcoin is the de facto currency, we're going to be using a lot of energy to maintain the network, energy that could be put to better use than just helping the network maintain itself.

The problem is that proof-of-work is the only known Sybil-prevention system thus far. Consensus research is still ongoing and has not stopped with proof-of-work, but for now it's the best that we have. In terms of up-and-coming competitors to proof-of-work, there is a big one that comes to mind: *proof-of-stake*. Proof-of-stake isn't perfect, either, but it can complement proof-of-work.

Proof-of-stake is a consensus mechanism that relies instead on computational power to prevent Sybil attacks on stake in the network. Usually, by stake it means amount of cryptocurrency owned by the miner. The idea is that the more cryptocurrency you have, the more invested you are in ensuring the stability of the network and the less likely you are to perform a 51 percent attack to fork the blockchain. Delegated proof-of-stake is an innovation of proof-of-stake where a set of 101 delegates can vote on block generators. Both delegated proof-of-stake and proof-of-stake are still undergoing research, but if either proves to be secure in the long term, they could be used to complement or maybe even completely replace proof-of-work.

Defining the Terms

So why the term dapp? Why decentralized app? Why not Decentralized Application Organizations or Decentralized Autonomous Corporations or Decentralized Application?

The cryptocurrency space is saturated with differentiating terms for this theoretical and partially implemented ecosystem of dapps. The best way to dive into why I've chosen the term dapp is to dive into all of the existing terms for dapps and see what they're all about. Let's begin with dapp itself.

Decentralized applications (DAs)

Decentralized Applications is the name of this book. I could've just as easily chosen DO or DAO or DAC. Why dapp? Because the common word in all of the phrases is "decentralized." Decentralized apps are the superclass of all decentralized entities that involve software.

Decentralized organizations (DOs)

A DO is one that empowers all of its employees. The term doesn't really apply to the tools the organizations use; it's more a description of how it's structured. There are varying degrees of decentralization, and complete decentralization isn't necessarily the best way of doing things. In a traditional organization, there is a rigid, hierarchical structure of command.

A decentralized organization gives voice to its employees and the power is spread more evenly among everyone. Company practices and milestones are made auditable by everyone and can be stored in a decentralized storage network for optimal resiliency. Humans don't need to be the only ones making decisions: smart contracts can take on roles like paying people by a certain date. DOs don't need to be based in a certain city, either; members can be spread out globally. In some systems (for example, Bitcoin), collusion is seen as a bug. In a decentralized organization, collusion is a feature. In the political realm, we call decentralized power democracy. We're seeing some startups recently opt for a more decentralized structure, especially as remote collaboration tools like Slack and GitHub progress.

Automated agents (AA)

AAs don't need to mean SkyNet or some general artificial intelligence. We've had automated agents for at least a decade. AA just means a piece of software that runs without any human intervention; in other words, autonomously. A perfect example would be a computer virus. The developer made it and released it to the wild. It then decides to self-replicate or carry out any other maintenance algorithm with which it was encoded. Another example would be a daemon. A daemon is a program that runs as a background process in an operating system, like an email program. Automated agents have their ups and downs, they don't require any maintenance, but having unchecked agents can lead to an uncontrollable source of possible danger for humanity—more on that in Chapter 6.

Decentralized autonomous organizations (DAOs)

This was actually what I was originally intending on calling the book before switching over to dapps. DAOs are just like DOs except AI makes the decisions, not humans. The protocol lives in a decentralized stack and doesn't heed any legal bindings. Humans aren't in charge, they are on the edges. AI is what makes the decisions and the DAO maintains itself. DAOs aren't just defined by having AI make all the decisions, they also have their own internal capital.

In short, each of these is a subclass of dapps, and a DAO is a dapp with AI controlled decisions and humans on the edges. Collusion isn't treated as a feature as in decentralized organizations but instead as a bug. Bitcoin is an example of a DAO.

Decentralized Autonomous Corporations

This one is controversial. Some think that this shouldn't even be a phrase because the word corporation is derived from the legacy system of legal contracts and hierarchical centralized control from which we are trying to evolve. The other side of the argument is that a DAC is a subclass of a DAO that pays dividends to its members. I am going to side with the former argument because I don't like the term corporation and if a DAO wants to implement dividends to its human and/or machine members, it can as a DAO, not a DAC.

So we've talked about dapps, DOs, DAOs, AAs, and DACs with an example of each. Let's take a look at Figure 1-3 to help make things a little clearer.

I like this chart a lot because it puts into context everything we've been talking about thus far. We're not at a stage yet where we can make AI (the holy grail, as the chart puts it), but we are at the next stage of evolution where we can begin making DAOs.

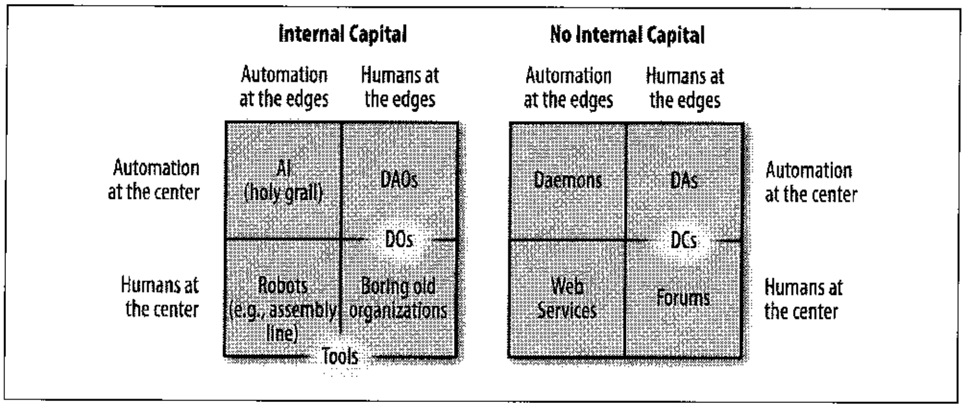


Figure 1-3. Types of organizations (Credit: Vitalik Buterin)

For brevity's sake, we're going to be using the term dapp throughout the book. Because dapps are the superclass of all decentralized software, and I'm going to be discussing different tools you can use as well as methodologies to define your dapp, you are best suited to decide what type of dapp you want to create.

My definitions have been pulled from my research from the cryptocurrency community, and my aim isn't to put yet another label on concepts or to create new paradigms. In fact, my aim is to simplify the space as much as possible such that you can fully grasp all the tools at your disposal to create a profitable decentralized app. The centralized app space has been nearly exhausted of ideas and it's time to iterate again after seeing its pros and cons. Dapps are the next wave of software and hopefully this book will prepare you to be a part of it.

Getting Started

I hope I've given a sufficient introduction as to what a decentralized application is. Much is still to be explained but this should've given you a brief introduction to the space and all the terms and acronyms associated with a dapp. My aim for this book is to first give you an explanation of dapps, what they are, why to build them, and what a thriving dapp ecosystem looks like. Then, I'll explain ways that you can implement your own using tools that currently exist. Finally, we'll take a deep dive into a few major players in the dapp space.